

Understanding the Impact of GitHub Suggested Changes on Recommendations Between Developers

Chris Brown, Chris Parnin
North Carolina State University

論文情報

Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference
and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020).
Association for Computing Machinery, New York, NY, USA, 1065–1076.

紹介者：堀口 日向

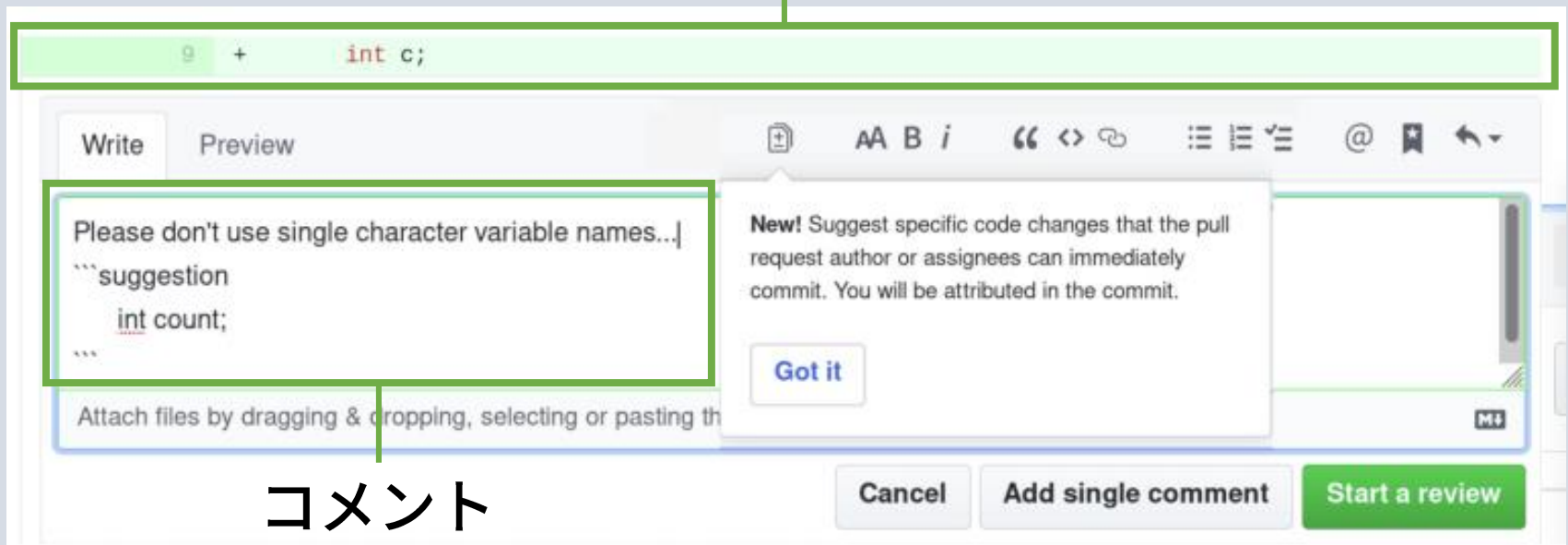
- ソフトウェアエンジニアにとって
ピアレビューやペアプログラミングは，非常に有益
 - エンジニア個々人のスキルアップ[3, 17]
 - 新しい開発ツールの採用[44]
 - コードの理解を改善[36]
- GitHubのPull Request (PR) 機能は，
外部の開発者がOSSプロジェクトに貢献する方法の1つ
⇒ 開発者が互いにコードの改善案などを提案できる場

投稿されたPRに対して改善案などを提案できる機能 (1/2)

レビューコメント (Review Comment)

PR内のコードに対して、テキストでコメント

ソースコード



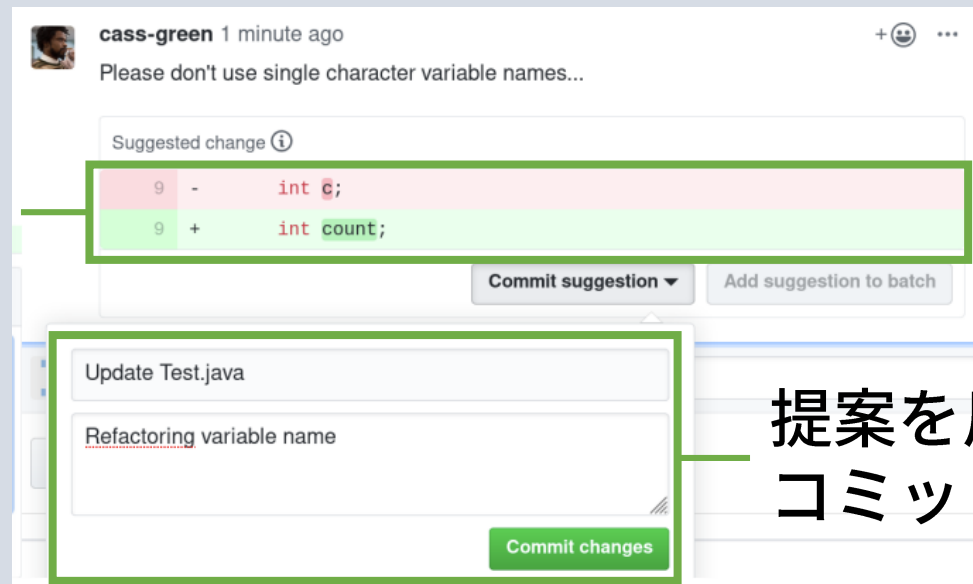
コメント

投稿されたPRに対して改善案などを提案できる機能 (1 / 2)

コード提案 (Suggested Changes)

PR内のコードに対して、追加のコード変更を提案
PR投稿者は、提案されたコードを、
そのままPR内のコードに対して反映可能

コード変更の
提案内容



提案を反映するための
コミット機能

目的とリサーチクエスト

GitHubのPR上でのコードの改善案の提案が
開発プロセスに与える影響を明らかにし、

目的

ピアレビューやペアプログラミングのための
システムの有効性を高めるための示唆を得る

リサーチクエスト

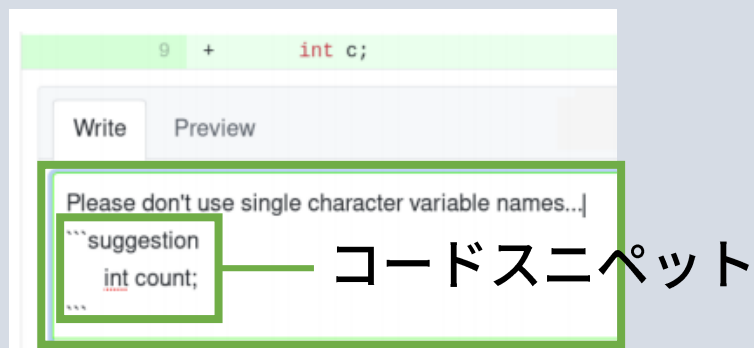
- RQ1：PR内のコードに対してどのような種類の提案が行われるか？
- RQ2：提案は開発プロセスに対してどの程度効果があるか？
- RQ3：提案はPRに対してどのような影響を与えるか？
- RQ4：提案は開発者にとってどのように役立っているのか？

データセット

対象期間：2018年10月以降に更新されたPR
(コード提案機能がリリースされた後)

対象PR数：約15万件

- 各PRにコード付きレビューコメントまたはコード提案が存在するかを確認
 - コード付きレビューコメント (以降レビューコメント)



⇒ 計22プロジェクト (例：nodejs/node, rust-lang/rust, ...)

- コード提案が存在するPR：約1.7万件
- レビューコメントが存在するPR：約7千件

PRに対してどのような種類の提案が行われるかを定性的に分析

1. データセットから100個のPRをランダムサンプリング
2. レビューコメントとコード提案を合わせて目視
3. オープンコーディングによるカテゴリ化

⇒ 4つのカテゴリを特定

修正 (Corrective)

機能に影響する問題の修正を提案

• 例

Suggested change ⓘ		
-		<code>`(function(){__BUILD_MANIFEST = JSON.parse('\${clientManif</code>
+		<code>`(function(){self.__BUILD_MANIFEST = JSON.parse('\${client</code>

PR投稿者が代入先変数の間違い

レビュアーがselfキーワードを追加する修正を提案

フォーマット (Formatting)

コードの見た目のみに影響する提案

• 例

Suggested change ⓘ		
-		<code>for i , j in product(range(-10,10), (0,20)):</code>
+		<code>for i , j in product(range(-10, 10), (0, 20)):</code>

レビュアーがコードのスペースを改善することを提案

改良 (Improvement)

リファクタリング，最適化の提案

• 例

Suggested change ⓘ

```
await Promise.all(manifests.map(x => makeManifest(reporter, x)))  
await Promise.all(manifests.map(manifest => makeManifest(reporter, manifest)))
```

コードの可読性を改善するために，レビュアーが変数名の変更を提案

非機能 (Non-Functional)

ドキュメントやコメントのスペル・文法の修正など，
コードに影響を与えない提案

• 例

Suggested change ⓘ

```
When we load the settings, we'll do it in two stages. First, we'll deserialie th  
When we load the settings, we'll do it in two stages. First, we'll deserialize
```

レビュアーがスペルミスの修正を提案

各カテゴリの提案された割合

	<i>n</i>	Percentage
Non-Functional	36	36%
Improvement	34	34%
Corrective	16	16%
Formatting	14	14%

⇒ 特に非機能的な提案や、リファクタリングの提案に効果的

RQ2：提案は開発プロセスに対してどの程度効果があるか？

提案の効果を，以下のメトリクスで分析

- 提案が存在するPRのマージ率
- 提案された改善案の受け入れ率
- 提案が存在するPRがマージされるまでの時間
- 提案が投稿されるまでの時間
- 提案された改善案が受け入れられるまでの時間

RQ2：提案は開発プロセスに対してどの程度効果があるか？

提案が存在するPRのマージ率

レビューコメントやコード提案が存在するPR/しないPRで
マージ率を比較

Pull Requests	<i>n</i>	Rate
with suggestions	6982	69.4%
without suggestions	44268	69.3%

マージ率はほぼ同じ

⇒ 提案はPRがマージされるかどうかには影響を与えない

RQ2：提案は開発プロセスに対してどの程度効果があるか？

提案された改善案の受け入れ率

提案後のPR内のコミット内容から，提案されたコードを検索

⇒ ヒットしたら受け入れられたと判定

コード提案とレビューコメントのそれぞれの受け入れ率を比較

Type	<i>n</i>	Rate
suggested changes	17712	59.6%
review comments with code	6937	0.9%

⇒ 提案自体は，レビューコメント機能よりも
コード提案機能の方がPRに反映されやすい

RQ2：提案は開発プロセスに対してどの程度効果があるか？

提案が存在するPRがマージされるまでの時間
レビューコメントまたはコード提案を含んでいるPRと
含んでいないPRのマージされるまでの時間[日数]を
マンホイットニーのU検定 (有意水準5%) で比較

Type	<i>n</i>	Average	Median
with suggestions	6982	16.4	5.0
without suggestions	44268	6.4	1.1

提案を含んでいるPRの方が，時間がかかる (有意差あり)
⇒ 提案を受け入れる作業や，必要な議論が増えるため

RQ2：提案は開発プロセスに対してどの程度効果があるか？

提案が投稿されるまでの時間

PRに対して，レビューコメントまたはコード提案が投稿されるまでのそれぞれの時間[日数]を，マンホイットニーのU検定 (有意水準5%) で比較

Type	<i>n</i>	Average	Median
suggested changes	17712	10.5	0.7
review comments with code	6937	14.6	1.9

コード提案の方が，投稿までの時間が短い (有意差あり)

⇒ レビューコメント機能よりも，
コード提案機能を使用して提案した方が，
迅速に提案を行える

RQ2：提案は開発プロセスに対してどの程度効果があるか？

提案された改善案が受け入れられるまでの時間
レビューコメントまたはコード提案が、
PR内に受け入れられるまでの時間[日数]を、
マンホイットニーのU検定 (有意水準5%) で比較

Type	<i>n</i>	Average	Median
suggested changes	17712	5.4	0.3
review comments with code	6937	8.0	0.7

コード提案の方が、改善案を受け入れるまでの
時間が短い (有意差あり)

⇒ レビューコメント機能よりも、コード提案機能を使用の方が、迅速に改善案を受け入れることができる

RQ3：提案はPRに対してどのような影響を与えるか？

分析方法

コード提案が存在するPR/しないPRで，PRの内容に関するメトリクスをマンホイットニーのU検定 (有意水準5%) で比較

- 閉じられるまでの時間[分]
- マージされるまでの時間[分]
- コミット数
- 変更行数
- 変更ファイル数
- レビューコメント数
- 関連するIssueのコメント数
- コメントした開発者数

RQ3：提案はPRに対してどのような影響を与えるか？

時間に関する結果（1/2）

メトリクス	コード提案の有無	中央値	有意差
閉じられるまでの時間[分]	あり	8190.43	あり
	なし	2489.12	
マージされるまでの時間[分]	あり	5900.27	あり
	なし	1776.32	

コード提案が存在するPRは、マージされて閉じられるまでに時間がかかる

⇒ コード提案が存在するPRは、閉じられるまでの開発プロセスがより複雑になっている可能性

RQ3：提案はPRに対してどのような影響を与えるか？

時間に関する結果（2/2）

メトリクス	コード提案の有無	中央値	有意差
コミット数	あり	4	あり
	なし	1	
変更行数	あり	133	あり
	なし	26	

コード提案が存在するPRは、実際にコミット数と変更行数が多い

⇒ コード提案はレビュー中の問題解決に役立つが、
その分レビュープロセスにより多くの時間がかかる

RQ3：提案はPRに対してどのような影響を与えるか？

コミュニケーションに関する結果

メトリクス	コード提案の有無	中央値	有意差
レビューコメント数	あり	7	あり
	なし	0	
関連するIssueのコメント数	あり	5	あり
	なし	3	
コメントした開発者数	あり	3	あり
	なし	2	

コード提案が存在するPRは、レビューコメント数、関連するIssueのコメント数、コメントした開発者数が多い
⇒ コード提案が、PR投稿者にフィードバックを提供し、PRの改善についての議論を促進するために有用である

RQ4：提案は開発者にとってどのように役立っているのか？

分析方法

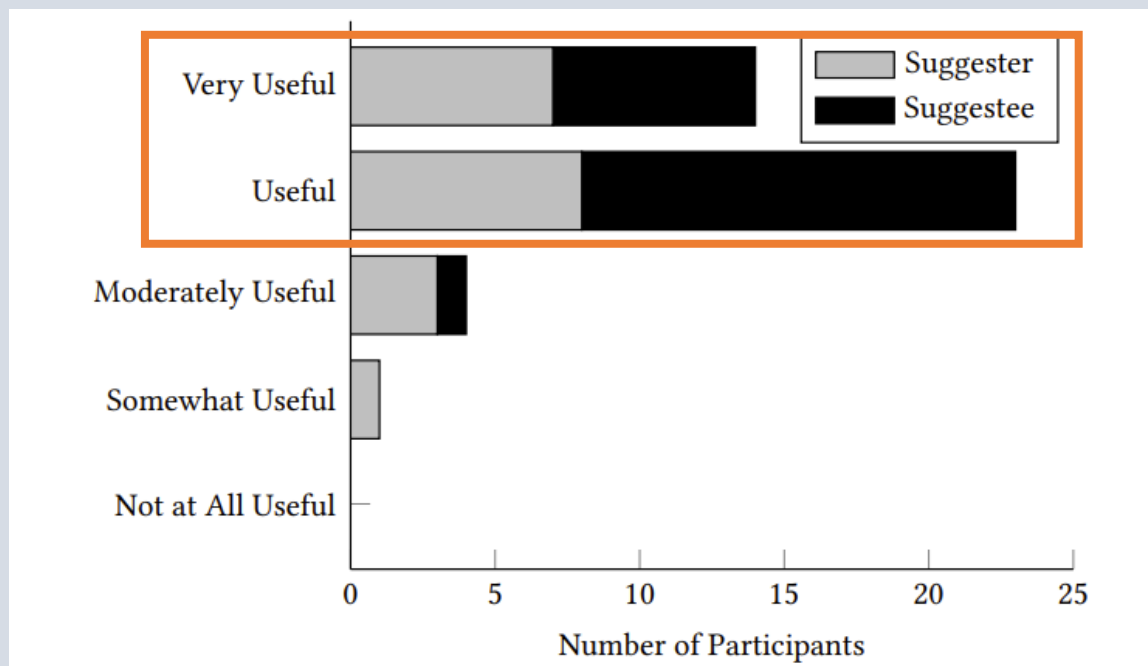
GitHubのコード提案機能についてアンケート

- ・ アンケート内容
 - ・ この機能の有用性を5段階のリッカート尺度で評価
 - ・ その評価をした理由について自由回答形式で質問
- ・ 対象者
 - ・ コード提案を行った開発者 (24人)
 - ・ コード提案を受け取った開発者 (19人)

580人中43人 (7%) が回答

RQ4：提案は開発者にとってどのように役立っているのか？

結果



79% (15人) の開発者が，有用または非常に有用

RQ4：提案は開発者にとってどのように役立っているのか？

結果

有用だと感じた理由 (特にレビューコメントにはない利点)

- 操作のしやすさ

PR投稿者は提案されたコードを**すぐPR内容に統合できる**

- 簡潔さ

テキストとして提案を書くよりも、**コードとして具体的な改善案を提案**した方が理解しやすい

- 速さ

テキストで提案してPR投稿者がコードを変更するのを待つよりも、**提案者自らがコードに変更を加える**方が早い

目的

ピアレビューやペアプログラミングのためのシステムの有効性を高めるための示唆を得る

アプローチ

GitHub上のPRのレビューコメントやコード提案を収集し、コードの改善案の提案によるPRへの影響を分析

結論

ピアレビューやペアプログラミングのためのシステムは、

- ・ **非機能的**な提案や、**リファクタリング**の提案に利用されやすい
- ・ レビュープロセスに**時間がかかるが、コードの改善には効果的**
- ・ **改善案を簡単に統合**できるようにすると効果が高まる